BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

EV324849404US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application for

METHODS AND APPARATUS FOR REMOTE PROCESS CONTROL

Inventors:

Richard L. Thibault
a citizen of the United States, residing at
90 Myrtle Street
Wrentham, Massachusetts 02903

Bruce S. Canna
a citizen of the United States, residing at
980 Maple Street
Mansfield, Massachusetts 02048

Gerald S. Couper
a citizen of the United States, residing at
71 South Road
Pepperell, Massachusetts 01463

Background of the Invention

The invention pertains to process control and has application to remote process control.

Process control refers to the control of the operational parameters of a system by monitoring one or more of its characteristics over time. It is used to insure that the quality and efficiency of the system remain within desired parameters over the course of time. While process control is typically employed in the manufacturing sector for process, repetitive and discrete manufactures, it also has wide application in service industries, such as environmental control.

Process control equipment typically utilizes control/sensing devices that are physically integrated into the systems being controlled. For example, a thermostat is typically used in environmental control to insure that building temperatures remain within specified parameters. Likewise, flow control sensors and automated valves are typically used in process manufacturing to insure proper fluid flow volumes.

Though in early process control systems, control/sensing devices were typically stand-alone units, modern process control systems provide central workstations for monitoring and controlling the control/sensing devices. Particularly robust systems are the I/A Series™ industrial automation systems designed, manufactured and marketed by the assignee hereof, The Foxboro Company, of Foxboro, Massachusetts, USA. In these systems, multiple control/sensing devices are coupled by way of buses to control stations which, in turn, are coupled by way of a local area network (LAN) to one or more operator workstations.

The I/A Series systems are built around the client/server model. Client applications software executing on the workstations exchange information with the control/sensing devices via a server, referred to as the "object manager," executing in distributed fashion in the control stations. Upon request by a client application, the server creates, locates, accesses and updates data structures ("objects") storing information on the status of at least selected control/sensing devices. For example, a client application that displays temperatures sensed by a thermocouple requests that the server create an object storing a temperature reading from the thermocouple and that the server notify the client each time the temperature changes.

Although modern process control systems, such as the I/A Series systems, have proven quite successful, to date they have provided only limited remote access capabilities. Thus, while numerous operator workstations may reside within the factory or facility in which the control/sensing devices are disposed, it has traditionally proven difficult to access and control those devices outside those areas.

Remote access and control of processes is desirable for a number of purposes. A plant manager who is "on the road," for example, may wish to monitor the plant processes while travelling. By way of further example, the manufacturer of process control equipment may require remote access to a plant's control/sensing devices in order to provide technical support.

An object of this invention is to provide improved methods and apparatus for process control.

Another object of the invention is to provide such methods and apparatus as permit monitoring and control of remote processes.

Still another object of the invention is to provide such methods and apparatus as can be readily adapted to existing automated process control systems.

Yet still another object of the invention is to provide such methods and apparatus as can be implemented without undue expense and without undue consumption of resources.

Summary of the Invention

The aforementioned objects are among those attained by the invention, which provides, in one aspect, a system for process control comprising a server digital data processor and a client digital data processor that are coupled by a network, such as the Internet or an Intranet. The server digital data processor, which is additionally coupled to a control/sensing device and associated interface equipment (collectively, referred to as "process control apparatus"), includes a command processor that transfers information between the network and the process control apparatus.

The client digital data processor includes an information client (e.g., an Internet web browser) capable of requesting and receiving an applet from the server digital data processor. That information client, further, defines a hardware-independent and operating system-independent virtual machine environment within the client digital data processor.

The client digital data processor executes, within that virtual machine environment, an applet that configures the client digital data processor as a "process controller" that establishes communications over the network with the command processor and that monitors and/or controls the process control apparatus via those communications. The applet is intermediate or executable code that is suitable for interpretation or execution within the virtual machine environment and that is hardware-independent, operating system-independent and windows system-independent

In further related aspects, the aforementioned applet can be, for example, JAVA programming language bytecode, and the virtual machine environment can be that created by a JAVA-enabled web browser.

According to other aspects of the invention, the command processor in a system for process control as defined above provides services (i.e., "software services") for access and modification of information regarding the process control apparatus. These services can permit, for example, the creation of a data structure object that stores information about the process control apparatus and that associates a name with that object; the destruction of such an object; the accessing of information in such an object; the updating of information in such an object; the determination, from an object name, of the physical address of the object; and the notification of changes in information stored by the object. The process controller generates and transmits over the network to the command processor requests for such services in order to monitor and/or control the process control apparatus.

A further aspect of the invention provides a system as described above in which the process controller generates and transfers commands (e.g., requests for service) over the network to the command processor in order to effect a transfer from the command processor of information regarding a status of the process control apparatus. The command processor responds to those requests by generating information on the status of the process control apparatus and transferring it back to the process controller over the network. The process controller can, for example, generate a user display based on that information.

In a related aspect, the command processor responds to selected commands (i.e., requests for event-driving access) by notifying the process

controller of changes in the status of at least selected aspects of the process control apparatus. By way of example, where the process control apparatus includes a thermocouple, this aspect of the invention permits notification of the command processor whenever the thermocouple senses a change in temperature that exceeds a predetermined delta value.

Still further aspects of the invention provide process control systems as described above in which the server digital data processor includes an information server (e.g., a hypertext transfer protocol server). An information client (e.g., web browser) in the client digital data processor establishes communications with the information server over the network and receives therefrom a hypertext markup language (HTML) document referencing the applet. The web browser generates a user display of that document and, in response to a user command, transfers to the information server a request for the applet.

Yet still further aspects of the invention provide systems for process control in which a first digital data processor executes a JAVA applet within a virtual machine environment defined on the digital data processor. The applet configures the digital data processor to generate a message to invoke a method in connection with monitoring and/or controlling a process control apparatus. An object manager, which is in communication with the JAVA applet, responds to the message for invoking the method.

Other aspects of the invention provide methods for process control paralleling the operations of the systems described above.

These and other aspects of the invention are evident in the drawings and in the detailed description that follows.

Brief Description of the Drawings

A further understanding of the invention may be attained by reference to the drawings, in which

Figure 1 depicts a system for process control according to the invention; and

Figure 2 is an event trace diagram depicting messages that flow among the components of the system of Figure 1 in an embodiment for graphing tends in process control apparatus data values.

Detailed Description of the Illustrated Embodiment

Figure 1 depicts a system 10 for process control according to the invention. The system includes client digital data processors 12, 14 and server digital data processor 16. The digital data processors 12, 14, 16 are connected to one another via network 18.

Server digital data processor 16 is, additionally, coupled to process control apparatus 19a - 19e via bus/network structure 30 and control stations 23a - 23e, as shown. The process control apparatus include conventional control/sensing devices, which are shown in the illustration as flow control valves, and associated interface equipment, which are marked "FBM" in the illustration. The process control apparatus 19a - 19e are intended to represent any conventional control/sensing devices and interface equipment of the type conventionally used to monitor and control processes -- including, by way of non-limiting example, continuous, repetitive and discrete processes, and environmental control processes, among others.

As discussed below, control stations 23a - 23e include objects storing information that control, and reflect the status of, their associated process control apparatus 19a - 19e. The control stations 23a - 23e also execute object management software (marked "OM") that manage and oversee access to those objects. The control stations 23a - 23e are of the type conventionally used in a distributed process control architecture. Preferred such control stations are commercially available from the assignee hereof, The Foxboro Company, as part of its I/A Series industrial automation systems.

The digital data processors 12, 14, 16 comprise conventional digital data processing systems of the type commercially available in the marketplace. Though client digital data processors 12, 14 are illustrated as a portable computer and a personal digital assistant, respectively, those skilled in the art will appreciate that these may comprise other computing systems, such as desktop computers and workstations, as well. The digital data processors 12, 14, 16 may be coupled to the network 18 directly, as shown, or other networks (e.g., LANs and WANs), routers, or interface servers (not shown).

The network 18 comprises any conventional digital data processing network (e.g., LAN or WAN), cable television-based network, wireless network and/or any telecommunications-based network capable of supporting communications between server digital data processor 16 and client digital data processors 12, 14. The network 18 preferably comprises the global Internet and/or an enterprise-based Intranet supporting communications via the TCP/IP protocol (i.e., the current standard protocol of the Internet). Utilization of networks supporting this protocol is advantageous insofar as it permits the use of commercially available products (such as web browsers, discussed below) in components of the illustrated embodiment. Those skilled in the art will appreciate that the invention is applicable to networks supporting other protocols, as well.

The digital data processors 12, 14, 16 execute software that respectively configure them for communication over the network 18. For example, they execute protocol stacks and other software that permit them to establish and carry out communications utilizing the TCP/IP network protocol. In addition, they execute information client/server software that configures them to carry on high-level communications, particularly, over the Internet.

More particularly, in the illustrated embodiment, server digital data processor 16 includes information server 20 responsible for establishing communications over network 18 with information clients executing on the client digital data processors 12, 14.

The information server 20 is preferably a hypertext transfer protocol (HTTP) server capable of transferring markup language information and, particularly, hypertext markup language (HTML) documents, to the client digital data processors 12, 14. In alternative embodiments of the invention, information server 20 can comprise any other such server capable of supplying an applet to the client digital data processors 12, 14 in response to requests by them.

The information server 20 establishes communications with the client digital data processors 12, 14 and, particularly, their respective information clients in the conventional manner known in the art. Once communications are established, the information server transfers to the information client an applet that executes within the virtual machine environment and that monitors and/or controls the process control apparatus via communications with a command processor in the server digital data processor 16, as discussed below.

The client digital data processors 12, 14 include information clients 22, 24, respectively, that are responsible for initiating and conducting at least preliminary communications with the server digital data processor 16 over the network 18. The information clients 22, 24, particularly, (1) initiate communications with the information server 20 over the network, (2) request and receive from the information server 20 an applet, and (3) define a platform-independent (i.e., a hardware-independent, operating system-independent and window system-independent) virtual machine environment within the respective

client digital data processor 12, 14. Such information clients are, in one emodiment, JAVA-compliant web browsers including the HotJava browser from Sun MicroSystems, Inc., NetScape Navigator from Netscape Communications Corporation, and the Internet Explorer from Microsoft Corporation.

As used herein, an applet is intermediate or executable code suitable for interpretation or execution within the virtual machine environment and that is hardware-independent, operating system-independent and windows system-independent. Preferred applets are in the form of Java bytecode of the type generated by the Java language compiler available from Sun Microsystems, Inc.

The aforementioned preferred web browsers define a preferred virtual machine environment comprising the Java programming language run-time platform and Java interpreter.

Although a preferred information client is a web browser, the invention can be practiced with other information clients capable of (1) initiating communications with the information server 20, (2) requesting and receiving from the information server 20 an applet, and (3) defining a platform-independent (i.e., a hardware-independent, operating system-independent and windows system indepent) virtual machine environment within the respective client digital data processor 12, 14 for execution of such an applet.

In addition to information server 20, server digital data processor 16 includes command processor 25, comprising front end 25a, interface section 25b, and an object manager 25c. Together, these transfer information between the network 18 and process control apparatus 19a - 19e. As shown in the illustration, the object manager functionality is distributed among the control

stations 23a - 23e. Each object manager maintains the data structures -- to wit, objects -- that control and reflect the status of its associated process control apparatus 19a - 19e.

The object manager 25c provides software services for access that permit the creation of named objects; destruction of such objects; accessing and updating of information in the objects; the locating of objects within the distributed process control architecture; and notification of changes in the information stored in objects (i.e., event-driven notification).

As noted, the object manager 25c allows uniquely named objects to be distributed over the control stations 23a - 23e in a location-independent way. Using the object manager 25c (via front end 25a), applets 26, 28 may create, read, write, and destroy instances of objects, which are subtyped into four categories: *variable* - used to contain an instance of any scalar data type (e.g., *int, float, etc.*) or a string; *alias* - used to contain a string which refers to the name of another object; *device* - used to identify a station or device in the system. An instance of a device type object contains no explicit state - the name of the object is itself the state; and *process* - used to identify an executing process in the system. A process object is identical to a device object in that there is no explicit state.

As indicated above, in order to manipulate instances of objects, the object manager 25c provides life cycle services, access services and connection services. Life cycle services are used to create, name, and destroy shared objects; to register the name of process-control objects; and to find the location of any object. Access Services are used to get and set the value of one or more

process-control and/or shared objects. Typically, access services are suitable for situations where a single transfer of data is sufficient.

Connection services are also used to get and set the value of one or more process-control and/or shared objects. However, these services are more suited for situations where multiple transfers of data are expected. In addition, connection services provide the ability for a client to be continuously updated with the value of an object when it exceeds a specified delta.

The object manager 25c relies upon the use of broadcasts over bus structure 30 in order to perform the above services. For example, when an applet 26, 28 makes an access request on an object by name, the object manager 25c will broadcast the access request to all stations 23a - 23e, if the object manager 25c does not know the location of object. Each station 23a - 23e then determines if it is the one that hosts the requested object. Only the station that hosts the named object responds to the request.

A preferred object manager 25c is that commercially available from the assignee hereof, The Foxboro Company, as part of its I/A Series of industrial automation systems. A software interface, or "API," of that preferred object manager is described in publicly available documentation, including the document entitled "Object Manager Calls," a copy of which is filed as an appendix herewith.

The command processor front end 25a executes on server digital data processor 16, configuring it to respond to requests from applets 26, 28 to establish communications with them over the network 18. Once communications are established, the front end 25a responds to requests received from applets 26,

28 over network 18 to transfer information to and from process control apparatus 19a - 19e via the object manager 25c.

Particularly, the front end 25a responds to requests received over the network in TCP/IP protocol to generate calls to object manager 25c in accord with its aforementioned API. Moreover, the front end 25a responds to information generated by the object manager 25c in response to those calls by transmitting that information back over the network 18, in accord with the TCP/IP protocol, to the applets 26, 28. In a preferred embodiment, the front end 25a presents a simplified interface to the object manager 25c, e.g., permitting applets 26, 28 to make requests and receive responses in the form of text strings, as discussed below.

Software implementing a preferred front end 25a as a Java programming language application is filed as appendix hereto. Those skilled in the art will appreciate that alternate embodiments may implement the front end in other programming languages suitable for, or that can be adapted to, provide an interface between the network 18 protocol and the object manager 25c.

Interface section 25b provides a software interface between the front end 25a and the object manager 25c. As noted above, in a preferred embodiment, the front end 25a is implemented as a Java programming language application. The object manager 25c, on the other hand, is implemented as a C programming language application and, accordingly, its API includes pointer-based parameters. The interface section 25b compensates for the inability of the Java front end 25a to utilize pointer-based parameters, e.g., by converting them to arrays as discussed further below.

Software implementing a preferred interface section 25b in the C programming language is filed as appendix hereto. Those skilled in the art will appreciate that interface section 25b is optional and may be excluded in embodiments where the front end 25a can make calls directly to the object manager 25c.

The client digital data processors 12, 14 execute applets 26, 28 within the virtual machine environments defined by the information clients 22, 24. Each applet 26, 28 configures its respective client digital data processors as a process controller that establishes communications over the network 18 with the command processor front end 25a and that monitors and/or controls the process control apparatus 19a - 19e via those communications. More particularly, the process controllers generate and transfer requests for service over the network 18 to the command processor 25 so as to effect the transfer of information controlling, and reflecting the status of, the process control apparatus 19a - 19e. The process controllers also receive information from the command processor 25, e.g., for display to an operator.

As noted above, the applets 26, 28 comprise intermediate or executable code that is interpreted or executed with in the virtual machine environment defined by the information clients and that is hardware-independent, operating system-independent and windows system-independent. Source code for preferred applets, in the Sun MicroSystems Java programming language, is provided in the appendix filed herewith.

A process control system constructed and operated in accord with system 10 of Figure 1 can be employed in a wide variety of process control embodiments. One such embodiment is shown in Figure 2 and described below.

That embodiment provides for generation, by an applet executing on the client digital data processor, of graphs showing trends in data values of process control apparatus coupled to a server digital data processor.

Figure 2 is an event trace diagram depicting messages that flow among the components of the system 10 of Figure 1 in the above-mentioned embodiment. The components of the system 10 are shown in the event trace diagram as vertical lines with the name of the component at the bottom of the line. Messages are represented by arrows. Each message flows in the direction of the arrow from component to component. Messages that happen earlier in time are toward the top of the diagram.

Referring to Figure 2, communication begins with the operator signalling the information client 22 to establish communications with the server digital data processor 16 over the network 18. The operator can signal the information client, e.g., a keyboard stroke or "mouse" click on the operator console (not shown). In the illustrated embodiment, the information client 22 is the Netscape web browser.

In response the operator's request, the information client 22 generates and transmits over network 18 a request for connection with information server 20, e.g., an HTTP server, executing on server digital data processor 16. Once the connection is established, the HTTP server 20 sends to the web browser 22 an HTML page that references (i.e., provides an address for) a trend-graphing applet. The HTML page also optionally includes text and graphics describing the applet.

The web browser 22 displays the HTML on the operator console. If the operator signals the web browser 22 that he or she wishes to access the applet, the web browser 22 transmits to the HTTP server 20 over the network 18 a request for the applet. It will be be appreciated that the applet may be transmitted to the web browser 22, along with an initial HTML document.

The HTTP server 20 responds to such a request for forwarding Java bytecode for the applet over the network 18 to the web browser 22. On receipt of the applet, the JAVA-compatible web browser 22 executes the applet 26 in the virtual machine environment defined in the web browser 22.

Once executing, the applet 26 sends a request to establish a separate communications link over the network 18 with the command processor front end 25a, e.g., a Java application executing on the server digital data processor 16. This separate connection is used by the applet 26 and the front end 25a to permit the exchange messages over the network and, particularly, to permit the applet 26 to make requests of the command processor 25 for process control apparatus data to be graphed.

Once communications are established, the applet 26, 28 generates a display on the operator console of the client digital data processor 12 and permits the operator to enter the names of process control apparatus data values (i.e., "points") that are to be graphed. On the operator's command, the applet 26 sends a request over the network 18 to the front end 25a specifying the OMOPEN service and listing the names of operator-specified points. The request is in text or ASCII format, e.g., "OMOPEN name1; name2; name3; etc."

On receipt of the OMOPEN request, the front end 25a creates a data structure required by object manager 25c, to wit an OM list, and includes in that data structure the names of the specified points. The front end 25a then makes

an "omopen list" call to the object manager 25c utilizing the aforementioned API. A further understanding of the OM list data structure and of the "omopen list" call, as well as the other data structures and calls to the object manager 25c, may be attained by reference to the appendix filed herewith.

The object manager 25c responds to the omopen list call by querying the respective process control apparatus 19a - 19e for current data values for the points. The object manager 25c returns those data values to the front end 25a which, in turn, generates and transmits to the applet 26, 28 a text message listing the initial data points. That message includes the keyword OMUPDATE, followed by the names and values of each of the points, e.g., "OMUPDATE point1=value; point2=value; etc." The applet 26, 28 graphs those initial data points on the operator console.

The object manager 25c then begins looping, while awaiting further requests from the client applet 26 and while awaiting updates on the data values from the object manager 25c. When such an update is received, the front end 25a generates and transmits to the applet 26 a further text message in the form "OMUPDATE point1=value; point2=value; etc." listing the updated data values points. The applet 26 graphs those initial data points on the operator console at the end of the graph time interval.

The front end 25a continues looping and forwarding updates until the operator signals the applet 26 to stop trend graphing. In that event, the applet 26 sends a close request over the network to the front end 25a in the form of a text message "OMCLOSE." On receipt of that request, the front end 25a, in turn, makes an omclose list call to the object manager 25c in accord with the aforementioned API. When that call returns, front end 25a sends an

"OMCLOSEOK" text message to the applet, 26 causing it to clear the trend graph.

At this point, the operator can either specify new points to the applet 26 or can tell the web browser 22, 24 to connect to a different information server. If the operator signals that he or she wishes to connect to another server, the client applet 26 breaks the connection with the server by sending an "OMBREAK" message to the front end 25a over the network. The front end 25a than resets, and waits for the next connection.

In a preferred embodiment, the method illustrated in Figure 2 is implemented in the Java programming language. As those skilled in the art will appreciate and as discussed above, all Java applets and Java applications run inside of a Java Virtual Machine. All implementations of the Java Virtual Machine are guarantied to be identical regardless of the many hardware platforms on which they run.

The above-described trend-graphing client Java applet preferably runs in the Java Virtual Machine that is implemented by Netscape Navigator version 2.02. The trend client applet 26 is intended to be portable. So it only uses those classes that are present in all implementations of the Java systems. The trend-graphing applet 26 uses Java system classes to manage the screen, and connect to the trend server, and provide timing intervals.

The trend-graphing applet 26 implements classes that conduct all operator interaction. For example, it accept the names of the points to be graphed. It also defines the GUI buttons used by the operator to signal when graphing is to start

or stop. Further, the trend-graphing applet 26 plots X-Y axes, graph the points, and parses messages from the front end 25a.

The applet 26 also processes the following messages from the server: "OMUPDATE name2 = value; name3 = value; ...."; OMCLOSEOK.

The illustrated front end 25 (or "trend-graphing server") is not portable to just any Java Virtual Machine because it must call outside of the Java environment to the object manager 25c. To do this, the trend server class is defined to have "native methods". A "native method" is any member function of a class that is implemented in a language other than Java. A native method can enable access to functions and data that are "native" to a particular hardware platform operating system or a running application (like the object manager 25c).

Native member functions are declared in the class as native. They are implemented in a library that is loaded by the Java environment at runtime. On Solaris this is a libfile.so file. On Windows NT this would be a library.dbl file. The native methods, which constitute the interface 25b, are defined to create a new OM list, add a named point to the list, open the list, check the list for updates (using dqchange), and close the list. Source code for a preferred implementation of native methods is supplied in the appendix filed herewith.

The command processor front end 25a runs in a Solaris implementation of the Java Virtual Machine. The front end 25a processes the following messages from the applet 26: "OMOPEN name1; name2; name3; ..." (in response to which it creates a list with the specified points and opens the list): "OMCLOSE" (in response to which it closes the list); and "OMBREAK" (in response to which reset and wait to accept a new connection).

Described above and illustrated in the drawings are improved methods and apparatus for process control. Those skilled in the art will appreciate that the embodiments discussed above and shown in the claims are merely illustrative and that other embodiments incorporating modifications within the reach of one of ordinary skill in the art fall within the scope of the invention, of which we claim: